Network service chaining with efficient network function mapping based on service decompositions

Sahel Sahhaf*, Wouter Tavernier*, Didier Colle*, Mario Pickavet*
*Ghent University - iMinds
Email: sahel.sahhaf@intec.ugent.be

***Abstract*—Network Service Chaining (NSC) is a service concept which promises increased flexibility and cost-efficiency for future carrier networks. The two recent developments, Network Function Virtualization (NFV) and Software-Defined Networking (SDN), are opportunities for service providers to simplify the service chaining and provisioning process and reduce the cost (in CAPEX and OPEX) while introducing new services as well. One of the challenging tasks regarding NFV-based services is to efficiently map them to the components of a physical network based on the services specifications/constraints. In this paper, we propose an efficient cost-effective algorithm to map NSCs composed of Network Functions (NF) to the network infrastructure while taking possible decompositions of NFs into account. NF decomposition refers to converting an abstract NF to more refined NFs interconnected in form of a graph with the same external interfaces as the higher-level NF. The proposed algorithm tries to minimize the cost of the mapping based on the NSCs requirements and infrastructure capabilities by making a reasonable selection of the NFs decompositions. Our experimental evaluations show that the proposed scheme increases the acceptance ratio significantly while decreasing the mapping cost in the long run, compared to schemes in which NF decompositions are selected randomly.**

I. Introduction

With the rise of Software-Defined Networking (SDN) and Network Function Virtualization (NFV), the service chaining trend has taken on new importance. These two new developments can significantly simplify the service chain provisioning. A service consists of several service functions, traditionally implemented by middleboxes. Traffic flows have to traverse these middleboxes in a given order. Service chain is an abstraction to define high-level services in a generic way. The service is described by a chain of high-level Network Functions (NF) and pre-defined parameters which is referred to as a Service Graph (SG). Several research projects have been started to investigate different aspects of service chaining and address the limitations. Examples are i) a dedicated working group (Service Function Chaining Working Group) in IETF investigating the service chaining architecture, ii) the Network Functions Virtualization (NFV) group within ETSI focusing on providing software-based telecommunications services to be run in virtualized environment rather than special purpose appliances, and iii) UNIFY, an EU-funded FP7 project focusing on developing an automated, dynamic service creation architecture based on a dynamic fine-granular service chaining model leveraging Cloud virtualization techniques and SDN.

In light of the virtualization trend, the Virtual Network Embedding Problem (VNEP) is the problem of mapping/embedding a virtual network to the components of a physical network. Figure 1 depicts the general idea of VNEP. Given multiple virtual networks and a common physical infrastructure, we search for an embedding which maps virtual nodes/links of virtual networks to nodes/paths in the physical network. For NFV-based services this problem is translated to mapping SGs to the physical infrastructure.

978-1-4799-7899-1/15/$31.00

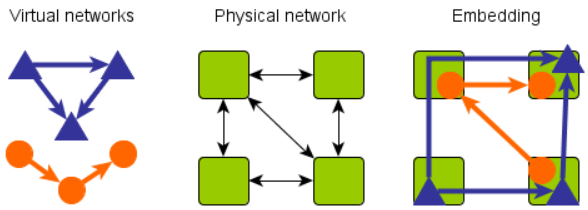

Fig. 1. Network embedding concept

As detailed in [1], there exist several algorithmic approaches in the literature to solve VNEP. These approaches can be categorized in different groups based on the envisioned setting. Mixed-Integer Programming is widely used in the optimal algorithms and may also yield polynomial-time heuristics [2], [3]. The heuristic solutions have been investigated abundantly in the literature. These solutions range from meta-heuristic approaches like ant-colony optimization [4] over graph isomorphism approaches [5] to Markov-Chain random walks [6]. Many of these algorithms are designed for a certain setting and are only evaluated on specific scenarios. This complicates the comparison of their performance in terms of acceptance ratio, network load, etc. and also time complexity.

Our contribution. In this paper, we propose an algorithm for VNEP taking NF decompositions into account. NF decomposition refers to converting an abstract NF (e.g. FireWall) to more refined NFs (e.g. an iptables- or OpenFlow-based FireWall) or decomposing the NF into several (potentially abstract) NFs interconnected into a graph with the same external interfaces as the higher level NF. Having different decompositions for an NF (see FireWall example), a service can be realized through multiple decomposition options. Selecting an NF decomposition without considering different aspects such as NF resource demands and the available resources in the infrastructure can degrade the efficiency of the embedding algorithm drastically. Therefore, in the proposed algorithm, service decomposition options are given as the input of the embedding algorithm and we propose a mechanism for decomposition selection to minimize the cost of the mapping. The algorithm maps the selected decomposition to the physical infrastructure in such a way that resource consumption is minimized. This is proportional to maximizing the number of

embedded service requests. The proposed scheme increases the acceptance ratio in the long-run while QoS requirements of the service requests are fulfilled.

Solving VNEP is NP-hard in most of the cases [7]. Additionally, the complexity of VNEP increases with the level of information provided. Giving the service decomposition options as input would increase the complexity of the VNEP significantly and thus, finding the optimal solution might not be an option for large-scale scenarios. Therefore, a heuristic-based approach is considered.

The performance of the proposed scheme is evaluated in a simulation environment and compared with scenarios where different metrics are used for decomposition selection. Based on the results, the efficiency of the embedding in terms of acceptance ratio and cost improves significantly if a reasonable service decomposition is selected.

Importantly, this paper is different from the other works in the sense that service decompositions are taken into account at the time of embedding and a resource-aware selection is made.

The rest of the paper is organized as follows. Section II details service decompositions. The problem and network model are described in Section III. The proposed algorithm is introduced in Section IV. Section V reports the performance evaluation results. Finally Section VI concludes the paper and discusses the future plans.

II. Service Decomposition

Service decomposition is the process of translating an SG containing high-level (compound) NFs to SGs with more elementary, implementation-close NFs, which can eventually be mapped onto the infrastructure. An abstract (compound) NF can be implemented through more refined NFs or it can be decomposed into multiple (potentially abstract) NFs interconnected in form of a graph with the same external interfaces as the higher level NF (see Figure 2). The objective of service decomposition is to enable: i) the re-use of elementary blocks (NFs), ii) building new, more complex services and iii) requesting high-level services without being concerned about the detailed implementations.

Assuming a high-level parental control NF, it can be decomposed to i) Traffic Classifier, ii) Web Proxy and iii) FireWall NFs. Each of these NFs can be implemented through more refined NFs, e.g. a FireWall can be realized by i) iptables-based FireWall or ii) OpenFlow-based Firewall. Traffic flows should traverse these NFs in a given order ('Traffic Classifier'→'Web Proxy'→'FireWall'). This can be represented by a graph and we refer to it as an NFG. In this approach there can be multiple decompositions for an NF. More formally, service decomposition is defined as a mapping of each NF into a set of NFGs: $NF_i \rightarrow \{NFG_1^i, NFG_2^i, ...\}$.

Figure 2 depicts an example service decomposition. The high-level NF2 is decomposed to NF4 and NF5. Each of them can be further decomposed to more elementary NFs. These decompositions are determined in design time of the NFs or SGs. They can be stored as a tree-like data structure in a database to be used by the embedding algorithm. The leaves of the decomposition tree must include NFs for which low-level implementation and resource requirements are available.

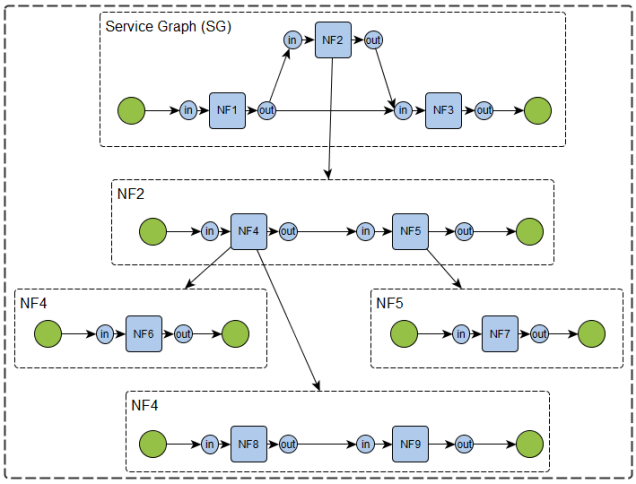

Fig. 2. Example of service decomposition

III. Problem Description and Network Model

Service requests arrive over time. The embedding algorithm should determine if the NFs and their connections in the SGs can be mapped to the infrastructure. It then assigns resources to the NFs which are released once the requests expire.

Given an SG, its elementary NFs can be of different types which means that they can be implemented through different techniques such as: i) Virtual Machine (VM) images in different virtualization techniques: Xen, Vmware, ii) process in a container iii) packet I/O drivers: DPDK, or iv) hardware appliances. This imposes a constraint on the embedding as not all physical nodes of the infrastructure support all types.

From technical point of view, interconnecting NFs of the same type (e.g. same-type processes in containers) is less complex compared to interconnection of different type NFs (e.g. a process in a container and a DPDK-based NF). Additionally, the more the number of NFs with the same type, the more NFs might be mapped into a same physical node. The latter leads to less resource consumption, if the similar-type NFs are interconnected directly (no NF with another type in between). Therefore, prioritizing service decompositions with more directly interconnected NFs of the same type improves the performance of the embedding in terms of resource consumption while enabling less complex NFs interconnections. To prioritize such decompositions, Cluster-Factor (CF) of each decomposition is calculated: the NFs with similar types which are connected (without intermediate NFs with other types) are grouped in a same cluster. The number of clusters in the decomposition is the CF of that decomposition. This factor is further used for the decomposition selection.

Our objective is to minimize the embedding cost which is achieved by minimizing the resources consumed in the infrastructure to map a request. This allows accepting more requests over time and increases the acceptance ratio. As service decompositions are known from the design time, we can make a resource-aware decomposition selection taking decompositions CFs into account at the time of the embedding.

Physical network. The physical infrastructure is modeled as an undirected graph. The infrastructure consists of nodes (N) connected via links (L). Each node has certain capacity

in terms of computation (C) memory (M) and storage (S), and links have delay (D) and capacity in terms of bandwidth ($BW-L$). Each node has the capacity of bandwidth ($BW-N$) that can go through the node. Each unit of resources has a cost and nodes can support different types of NFs. The parameters of the physical network infrastructure are listed below.

$$G_p = (N_p, L_p)$$
$$\forall u \in N_p C_u, S_u, M_u, BW-N_u, C_{cost_u}, S_{cost_u}, M_{cost_u} \in N^+$$
$$\forall u \in N_p \; u \in \{VM, process, I/O, hardware\}$$
$$\forall e_{uv} \in L_p \; D_{e_{uv}}, BW - L_{e_{uv}}, BW_{cost_{e_{uv}}} \in N^+$$

Service request. A service (SG) can be realized through multiple decomposition options.

$$\forall SG \; Decomp = \{dc_1, dc_2, ..., dc_n\}$$

Each decomposition dc is represented as a directed graph to support the dependency between elementary NFs. The NFs in dc are represented as nodes connected via directed links in the graph. Each NF has certain requirements in terms of computation (c), memory (m) and storage (s) and links connecting different NFs should meet certain requirement in terms of maximum allowed delay (d) and bandwidth (bw). Each decomposition has a CF as well.

$$G_{dc} = (N_{dc}, L_{dc})$$
$$\forall i \in N_{dc} \; c_i, s_i, m_i \in N^+$$
$$\forall i \in N_{dc} \; i \in \{VM, process, I/O, hardware\}$$
$$\forall e_{ij} \in L_{dc} \; d_{e_{ij}}, bw_{e_{ij}} \in N^+$$
$$\forall dc \in Decomp \; CF_{dc} \in N^+$$

Given an SG, its $Decomp$ and a physical network G_p, we search for an embedding which minimizes the resource consumption while fulfilling QoS requirements of SG. To this end, a resource-aware decomposition selection while taking the decompositions CFs into account is proposed.

IV. Proposed algorithm

The algorithm is based on a backtracking mechanism and it considers that requests arrive over time. It is composed of two phases: i) Decomposition selection and ii) Mapping.

Decomposition selection. Given the physical network G_p, the service SG and its $Decomp$, the CF of each service decomposition dc is calculated. Additionally, for each NF in each dc, the number of candidate physical nodes with sufficient capacities which can potentially host that NF is counted. We define parameter p to be the minimum number of candidate physical nodes for NFs of a dc. Consider a service decomposition composed of NF1 and NF2 which can be potentially mapped to 3 and 5 physical nodes respectively. Then $p = min\{3, 5\} = 3$. This parameter is defined to enable a resource-aware selection. We select a decomposition which is less restricting (has more mapping options). We define a cost function which combines CF, p and n (n is the number of NFs in a decomposition).

$$C(dc) = a.1/p_{dc} + b.CF_{dc} + g.n_{dc}$$

In the Decomposition selection phase, we select a decomposition with minimum cost $C(dc)$. The a, b and g parameters are introduced to tune the impact of different factors. Note that in addition to reduced embedding cost, a clever selection at this phase can avoid the backtracks in the mapping phase as well. Algorithm 1 provides the pseudo code of this phase.

DecompositionSelection($Decomp$)
Data: service decompositions $Decomp$
Result: minimum cost decomposition
$Cost$=[];
for $dc \in Decomp$ **do**
  CF_{dc}= number of clusters;
  p_{dc}= minimum (number of candidate physical nodes for NFs in dc);
  n_{dc}= number of NFs;
  $Cost(dc) = a.1/p_{dc} + b.CF_{dc} + g.n_{dc}$
end
select minimum($Cost$);

Algorithm 1: DecompositionSelection pseudo code

Mapping. We cluster the NFs of the selected decomposition based on their types and connections (see explanation for CF calculation) and sort the clusters and their NFs based on their requirements in descending order. We start mapping the NFs of the cluster with maximum requirement.

For each of the unmapped NFs in the sorted list, we sort its corresponding candidate physical nodes based on their distance (hop count) to the used physical nodes in ascending order. We select the physical node from this sorted list by which the links connected to the NF can also be mapped in the physical network. If such a physical node does not exist, the algorithm backtracks to the previous mapped node and checks the next candidate and repeats the same procedure. The pseudo codes of this phase are illustrated in Algorithms 2, 3, 4 and 5.

ServiceMapping (dc)
Data: selected decomposition dc
cluster dc;
$sortedNF$ =sort clusters and their NFs based on their requirements in descending order;
for $NF \in sortedNF$ **do**
  $candidate$= sort the candidate nodes for NF based on their distance to the used physical nodes;
  $i = 0$;
  while *MapNF(NF,candidate[i])==false* **do**
    $i+ = 1$;
    if $i \geq length(candidate)$ **then**
      backtrack to previous mapped NF and select the next candidate in its sorted list;
    end
  end
end

Algorithm 2: ServiceMapping pseudo code

V. Performance evaluation

The focus of our evaluations is on quantifying the advantage of considering service decompositions in terms of embedding cost and acceptance ratio. We compare different scenarios: i) embedding with minimum $C(dc)$ decomposition

selection and ii) embedding with random decomposition selection. We also evaluate the effect of a,b and g parameters in $C(dc)$ on the performance of the embedding.

A. Simulation experiments

The simulation environment is based on Python code. Libraries such as Networkx and Numpy are used for graph-based and numerical implementations. For the physical network we used "Interoute" topology from the Internet Topology Zoo[1]. Interoute is an international telecommunications service provider with the Europe's largest cloud services platform. The network is composed of 110 nodes and 148 edges. It is assumed that some of the nodes have general purpose servers supporting different virtualization technologies, and some of them have specific hardware appliances such as FireWall. This is selected randomly for nodes. The CPU, storage, memory capacity of nodes and bandwidth of links are numbers uniformly distributed between 100 and 300 and the cost of each unit of capacities is set to 1. The capacity of bandwidth that goes through each node is the sum of the bandwidth of links attached to the node. The delay of each link is a number with uniform distribution between 10 and 50 time units. The service requests arrive in a Poisson process with average rate of 4 requests per 100 time units. Each of them has exponentially distributed lifetime with an average of $\mu = 1000$ time units. The number of decompositions that each request can be realized with is a number uniformly distributed between 2 and 5 and the number of nodes in each decomposition is determined by a uniform distribution between 2 and 10. Each pair of nodes is connected with probability 0.5. The CPU, storage and memory requirements of an NF in a request are numbers with uniform distribution between 1 and 20. The NFs types are selected randomly. The maximum allowed delay for links in the requests is fixed to 1000 time units and the bandwidth requirement of links follows a uniform distribution from 1 to 50. Each simulation scenario is iterated 10 times.

We measure the average acceptance ratio and the relative embedding cost for service requests over time. We report the performance metrics against time to indicate how different approaches perform in the long run.

Figure 3 depicts the service acceptance ratio for the scenarios where only one of the 3 factors in $C(dc)$ is considered for decomposition selection. Such an evaluation gives a first idea on how to tune the parameters (a, b and g) to achieve a more efficient embedding. As we can observe, considering the number of NFs (g=1) results in higher acceptance ratio compared to the scenarios where only CF (b=1) or p (a=1) are considered. The embedding cost is another metric which should be evaluated to enable an efficient mapping.

Figure 4 reports the average embedding cost for the previous scenarios. As expected, selection of decompositions with less number of NFs results in lower cost compared to the other two scenarios. Therefore, to achieve an efficient embedding, in the $C(dc)$ function we tune the parameters in such a way that p and the CF have the same priority ($a = b = 0.25$) while the number of NFs (n) has higher priority ($g = 0.5$) in selection of the decomposition.

[1] http://www.topology-zoo.org/

MapNF(NF, $pnode$)
if *not enough capacity in pnode to map NF* **then**
  return false;
end
$success = CheckLinks(NF, pnode, mappedNodes)$;
if $success == true$ **then**
  $mappedNodes+ = (NF, pnode)$;
  $CPU_{pnode} - = CPU_{NF}$;
  $Memory_{pnode} - = Memory_{NF}$;
  $Storage_{pnode} - = Storage_{NF}$;
  return true;
end

Algorithm 3: MapNF pseudo code

CheckLinks(NF, $pnode$, $mappedNodes$)
for $l \in$ *links attached to NF* **do**
  if $neighbor\ attached\ to\ l \in mappedNodes$ **then**
    n= the physical node that the neighbor is mapped to;
    while *true* **do**
      $path$=shortest path between n and $pnode$;
      if $path == Null$ **then**
        return false;
      end
      $success = CheckQoS(path)$;
      if $success == true$ **then**
        for $link \in path$ **do**
          $BW_{link} - = BW_l$
        end
        for $node \in path$ **do**
          $BW_{node} - = BW_l$
        end
      end
    end
  end
end
return true;

Algorithm 4: CheckLinks pseudo code

CheckQoS($path$)
for $node \in path$ **do**
  if $BW_{node} < BW_{virtuallink}$ **then**
    remove the node from the network graph;
    return false;
  end
end
for $link \in path$ **do**
  $EndtoEndDelay+ = delay_{link}$;
  if *EndtoEndDelay exceeds the constraint* **then**
    remove the link from the network graph;
    return false;
  end
  if $BW_{virtuallink} > BW_{link}$ **then**
    remove the link from the network graph;
    return false;
  end
end
return true;

Algorithm 5: CheckQoS pseudo code

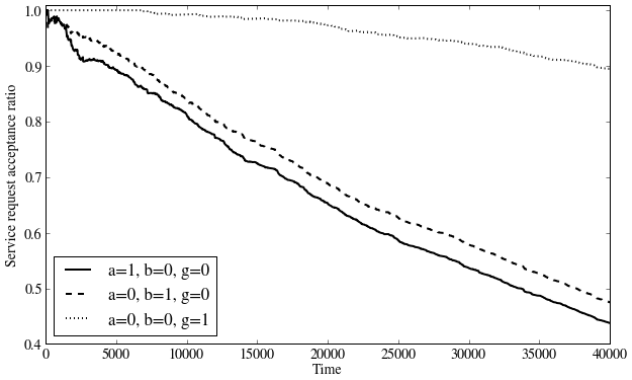

Fig. 3. Service request acceptance ratio over time

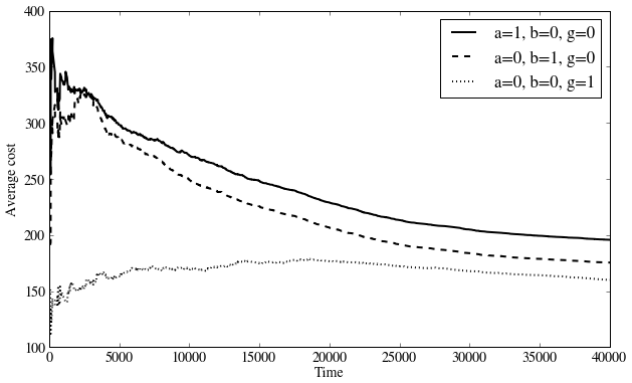

Fig. 4. Average cost of accepting requests over time

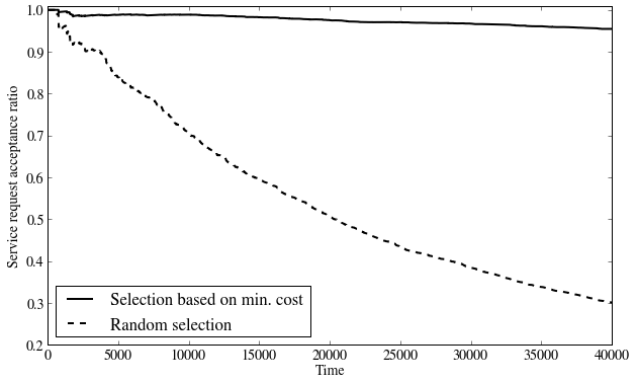

Fig. 5. Service request acceptance ratio over time

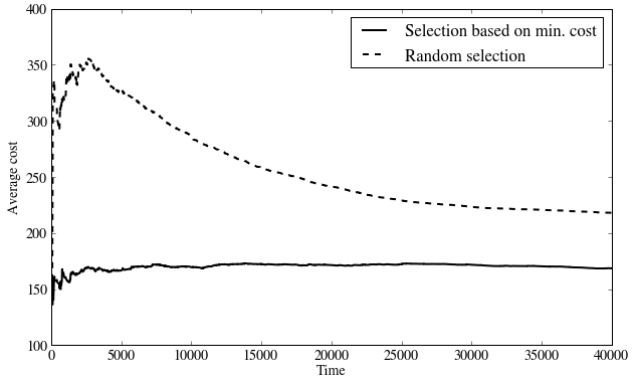

Fig. 6. Average cost of accepting requests over time

In the next experiment, we evaluate the scenario in which a decomposition is selected following the explained values of $a = 0.25$, $b = 0.25$ and $g = 0.5$ in $C(dc)$ and compare it with a scenario in which a decomposition is selected randomly. As we see in Figures 5 and 6, ignoring the network resources, CF and number of NFs in selecting a decomposition can impact the service acceptance ratio and embedding cost drastically while a promising performance can be achieved by a reasonable selection of decompositions.

VI. Conclusion

Knowing that a high-level Network Function (NF) can be decomposed in multiple ways to several less abstract NFs, a Network Service Chain (NSC) composed of NFs can be realized through multiple options referred to as service decompositions. In this paper, we proposed an algorithm to take such decompositions into account at the time of the embedding to provide an efficient placement of the NFs. The algorithm minimizes the resources consumed to map a service request. This allows accepting more requests over time. The simulation results indicated that resource-aware decomposition selection improves the performance in terms of request acceptance ratio and embedding cost significantly compared to a scenario in which decompositions were selected randomly.

In order to have a thorough evaluation, as part of the future work, we will consider more specific/realistic service requests instead of synthetic ones for the performance evaluation of the proposed scheme. Also it is important to know how the proposed scheme performs compared to the optimal solution. Therefore, as a next step, we will focus on evaluation of the scheme compared to the optimal solution (ILP formulation).

Acknowledgment

This work was conducted within the framework of the FP7 UNIFY project, which is partially funded by the Commission of the European Union.

References

[1] A. Fischer, J. F. Botero, M. Till Beck, H. De Meer, and X. Hesselbach, "Virtual network embedding: A survey," *Communications Surveys & Tutorials, IEEE*, vol. 15, no. 4, pp. 1888–1906, 2013.

[2] N. M. K. Chowdhury, M. R. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *INFOCOM 2009, IEEE*. IEEE, 2009, pp. 783–791.

[3] M. Rost, S. Schmid, and A. Feldmann, "Its about time: On optimal virtual network embeddings under temporal flexibilities," 2014.

[4] I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann, "Vne-ac: Virtual network embedding algorithm based on ant colony metaheuristic," in *Communications (ICC), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1–6.

[5] J. Lischka and H. Karl, "A virtual network mapping algorithm based on subgraph isomorphism detection," in *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*. ACM, 2009, pp. 81–88.

[6] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, and J. Wang, "Virtual network embedding through topology-aware node ranking," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 2, pp. 38–47, 2011.

[7] R. McGeer, D. G. Andersen, and S. Schwab, "The network testbed mapping problem," in *Testbeds and Research Infrastructures. Development of Networks and Communities*. Springer, 2011, pp. 383–398.